

Politechnika Wrocławska

Klawiatura na ATmega32u4 z komunikacją I2C

Autor: Piotr Siembab 275418, Prowadzący: dr inż. Antoni Izworski, 16 Grudnia 2024

Spis treści

1 Wstęp

1.1 Wstęp teoretyczny

1.1.1 Klawiatury Mechaniczne

Klawiatury mechaniczne różnią się od membranowych przede wszystkim mechanizmem działania klawiszy. W klawiaturach mechanicznych każdy klawisz posiada oddzielny przełącznik mechaniczny (switch), który rejestruje naciśnięcie. Przełączniki te są znane z dłuższej żywotności, wyższej precyzji i lepszego feedbacku dotykowego w porównaniu do ich membranowych odpowiedników. Dzięki temu klawiatury mechaniczne cieszą się popularnością wśród programistów, graczy oraz entuzjastów technologii.

1.1.2 Technika Handwiringu

Handwiring to metoda budowy klawiatury polegająca na ręcznym lutowaniu każdego połączenia między przełącznikami a mikrokontrolerem. W odróżnieniu od gotowych płytek PCB, technika handwiringu pozwala na większą elastyczność w projektowaniu układu klawiszy, co jest szczególnie przydatne w niestandardowych projektach.

1.1.3 Klawiatury typu Split

Klawiatury typu split składają się z dwóch oddzielnych połówek, które mogą być ustawione pod dowolnym kątem i w dowolnej odległości od siebie. Taki układ ma na celu zwiększenie ergonomii pracy, umożliwiając naturalne ułożenie rąk i nadgarstków. Dzięki temu zmniejsza się ryzyko wystąpienia dolegliwości związanych z długotrwałym pisaniem, takich jak zespół cieśni nadgarstka.

1.2 Założenia Projektowe

Projekt klawiatury mechanicznej typu split zakłada stworzenie funkcjonalnego urządzenia o następujących cechach:

- **Układ Klawiatury:** Klawiatura ma mieć układ 5 rzędów i 7 kolumn, podzielony na dwie symetryczne połowy. Układ ten zapewnia ergonomię i wygodę użytkowania, umożliwiając naturalne ułożenie rąk podczas pisania.
- **Mikrokontroler:** Każda połowa klawiatury wyposażona jest w mikrokontroler Arduino Micro Pro z układem ATmega32u4. Dzięki wbudowanemu interfejsowi USB, mikrokontroler ten umożliwia łatwe podłączenie klawiatury do komputera oraz emulację urządzeń HID (Human Interface Device).
- **Wyświetlacz OLED:** Każda połowa klawiatury posiada wyświetlacz OLED, który może być używany do wyświetlania różnych informacji, takich jak aktywna warstwa, statusy lub inne dane użytkownika.
- **Komunikacja I2C:** Komunikacja pomiędzy mikrokontrolerami oraz wyświetlaczami OLED odbywa się za pomocą protokołu I^2C. Do połączenia połówek klawiatury wykorzystano kabel TRRS, który przesyła sygnały VCC, GND, SDA i SCL, umożliwiając synchronizację i wymianę danych.
- **Programowanie QMK:** Klawiatura jest zaprogramowana przy użyciu QMK (Quantum Mechanical Keyboard) Firmware. QMK oferuje szerokie możliwości konfiguracji, w tym mapowanie klawiszy, obsługę makr, tworzenie warstw oraz wsparcie dla komunikacji I^2C i wyświetlaczy OLED.
- **Obudowa 3D:** Obudowa klawiatury została zaprojektowana i wydrukowana na drukarce 3D, co umożliwia dostosowanie jej do indywidualnych potrzeb oraz zapewnia odpowiednią trwałość i estetykę urządzenia.
- **Funkcjonalność:** Klawiatura ma być w pełni programowalna, co pozwala użytkownikowi na dostosowanie jej do własnych preferencji i wymagań. Dzięki QMK możliwe jest tworzenie niestandardowych układów klawiszy, makr oraz innych funkcji.

Projekt ten ma na celu stworzenie nowoczesnej, ergonomicznej i wysoce konfigurowalnej klawiatury mechanicznej, która spełni oczekiwania wymagających użytkowników.

2 Omówienie technologii

2.1 Arduino Micro Pro

Arduino Micro Pro to kompaktowa wersja popularnej platformy Arduino, wyposażona w mikrokontroler ATmega32u4. Ten mikrokontroler charakteryzuje się wbudowanym interfejsem USB, co pozwala na łatwe podłączenie urządzenia do komputera i komunikację z nim bez potrzeby używania dodatkowych modułów konwersji USB-UART.

- **ATmega32u4:** Jest to 8-bitowy mikrokontroler AVR z rodziny mikrokontrolerów produkowanych przez firmę Atmel (obecnie Microchip Technology). Posiada 32 KB pamięci flash, 2.5 KB pamięci SRAM i 1 KB EEPROM. Jego wbudowany interfejs USB pozwala na emulację urządzeń HID (Human Interface Device), takich jak klawiatury i myszy, co jest idealne do projektów związanych z klawiaturami.
- **USB:** Interfejs USB w Arduino Micro Pro umożliwia łatwe programowanie i komunikację z komputerem. Dzięki wbudowanemu bootloaderowi można łatwo wgrać nowe programy bez potrzeby używania zewnętrznych programatorów. USB również zapewnia zasilanie dla mikrokontrolera, co upraszcza konstrukcję urządzeń.

2.2 Komunikacja I^2C

I^2C (Inter-Integrated Circuit) to protokół komunikacyjny, który umożliwia przesyłanie danych pomiędzy urządzeniami za pomocą tylko dwóch przewodów: SDA (dane) i SCL (zegar). Jest to protokół magistrali szeregowej, który pozwala na podłączenie wielu urządzeń do tych samych przewodów.

2.2.1 OLED

W przypadku sterowania wyświetlaczem OLED, Arduino Micro Pro działa jako master, a wyświetlacz jako slave. Master inicjuje komunikację, wysyłając sygnał start, a następnie adresuje slave'a, do którego chce wysłać dane. Po nawiązaniu połączenia, dane są przesyłane w formie bajtów. Wyświetlacze OLED często używają dedykowanych bibliotek, takich jak U8glib czy Adafruit SSD1306, które ułatwiają zarządzanie wyświetlaniem grafiki i tekstu.

2.2.2 Komunikacja między modułami

Klawiatura jest podzielona na dwie połowy, które komunikują się za pomocą kabla TRRS (Tip-Ring-Ring-Sleeve). Ten rodzaj kabla pozwala na przesyłanie czterech różnych sygnałów, które w tym projekcie są używane do przesyłania VCC, GND, SDA i SCL. Jedna połowa klawiatury działa jako master, a druga jako slave. Master wysyła sygnały zegarowe (SCL) i dane (SDA), a slave je odbiera. Komunikacja $I2C$ umożliwia synchronizację działania obu połówek klawiatury, przesyłanie informacji o naciśniętych klawiszach oraz sterowanie dodatkowymi komponentami, takimi jak wyświetlacze OLED.

2.3 Przełączniki w konfiguracji macierzowej z diodami

Konfiguracja macierzowa to sposób na efektywne podłączenie dużej liczby klawiszy do mikrokontrolera, minimalizując liczbę wymaganych pinów wejścia/wyjścia. Klawiatury mechaniczne korzystają z tej technologii, aby skanować stan każdego klawisza.

- **Matryca Klawiatury:** Klawisze są ułożone w układzie wierszy i kolumn. Każdy klawisz znajduje się na przecięciu wiersza i kolumny. Kiedy klawisz jest naciskany, zamyka obwód między odpowiednim wierszem a kolumną.
- **Dioda przy Każdym Przełączniku:** Każdy switch posiada diodę umieszczoną w taki sposób, aby przewodziła prąd tylko w jednym kierunku. Diody zapobiegają problemom z tzw. "ghostingiem" (rejestrowaniem fałszywych naciśnięć klawiszy) i "key rolloverem" (rejestrowaniem wielu jednoczesnych naciśnięć klawiszy).

Proces skanowania polega na sekwencyjnym aktywowaniu każdej kolumny i sprawdzaniu, które wiersze są połączone z aktywowaną kolumną, co pozwala określić, który klawisz został naciśnięty.

2.4 QMK Firmware

QMK (Quantum Mechanical Keyboard) Firmware to open-source'owe oprogramowanie, które umożliwia zaawansowaną konfigurację i programowanie klawiatur mechanicznych. QMK oferuje szereg funkcji, które znacznie rozszerzają możliwości standardowych klawiatur, takich jak:

- **Mapowanie Klawiszy:** Umożliwia przypisanie różnych funkcji do poszczególnych klawiszy, w tym tworzenie warstw, makr i specjalnych skrótów klawiszowych.
- **Obsługa Makr:** Pozwala użytkownikom na definiowanie sekwencji naciśnięć klawiszy, które mogą być wywoływane jednym przyciskiem.
- **Warstwy:** Umożliwia tworzenie wielu układów klawiszy, które można przełączać w zależności od kontekstu (np. warstwa numeryczna, warstwa z funkcjami multimedialnymi).
- **Komunikacja $\mathbf{I^2C}$:** QMK obsługuje komunikację między różnymi mikrokontrolerami za pomocą I^2C, co jest kluczowe w przypadku podzielonych klawiatur ze względu na łatwą implementację tej metody komunikacji.
- **Wsparcie dla Wyświetlaczy OLED:** QMK umożliwia sterowanie wyświetlaczami OLED, które mogą być używane do wyświetlania informacji takich jak warstwa aktywna, statusy czy inne dane użytkownika.

QMK Firmware jest szeroko stosowany w społeczności entuzjastów klawiatur mechanicznych, ponieważ oferuje ogromne możliwości personalizacji i optymalizacji klawiatury według indywidualnych potrzeb użytkownika.

3 Zestawienie materiałów

3.1 Główny wykaz części

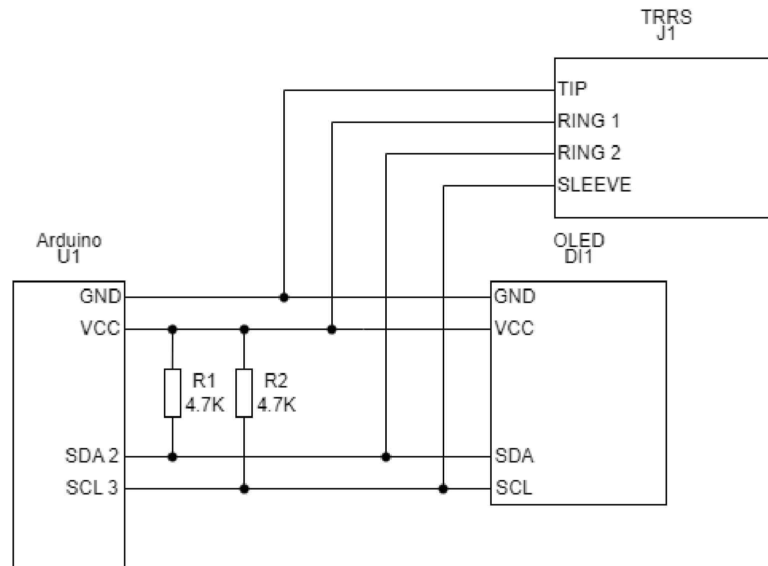
Komponent	**Szczegóły**	**Ilość**	**Jednostka**
Moduł Pro Micro USB-C	Atmega32U4	1	sztuk
Przełącznik klawiaturowy	Cherry MX Red	35	sztuk
Dioda prostownicza	1N4148	35	sztuk
Rezystor	4.7K	2	sztuk
Przewód elektryczny	-	25	sztuk
Wyświetlacz OLED	128x32 SSD1306	1	sztuk
Gniazdo TRRS	$3.5mm$	1	sztuk
Przewód TRRS-TRRS	male-male	1	sztuk
Srebrzony przewód miedziany	-	1.5	metr
Koszulka termokurczliwa	$\varnothing 1mm$	30	sztuk
Klawisz	-	35	sztuk
Zestaw obudowy	-	1	sztuk
Śruba M2 płaska	M2x8	4	sztuk
Śruba M3 płaska	M3x10	6	sztuk

3.2 Wykaz części wchodzących w Zestaw obudowy

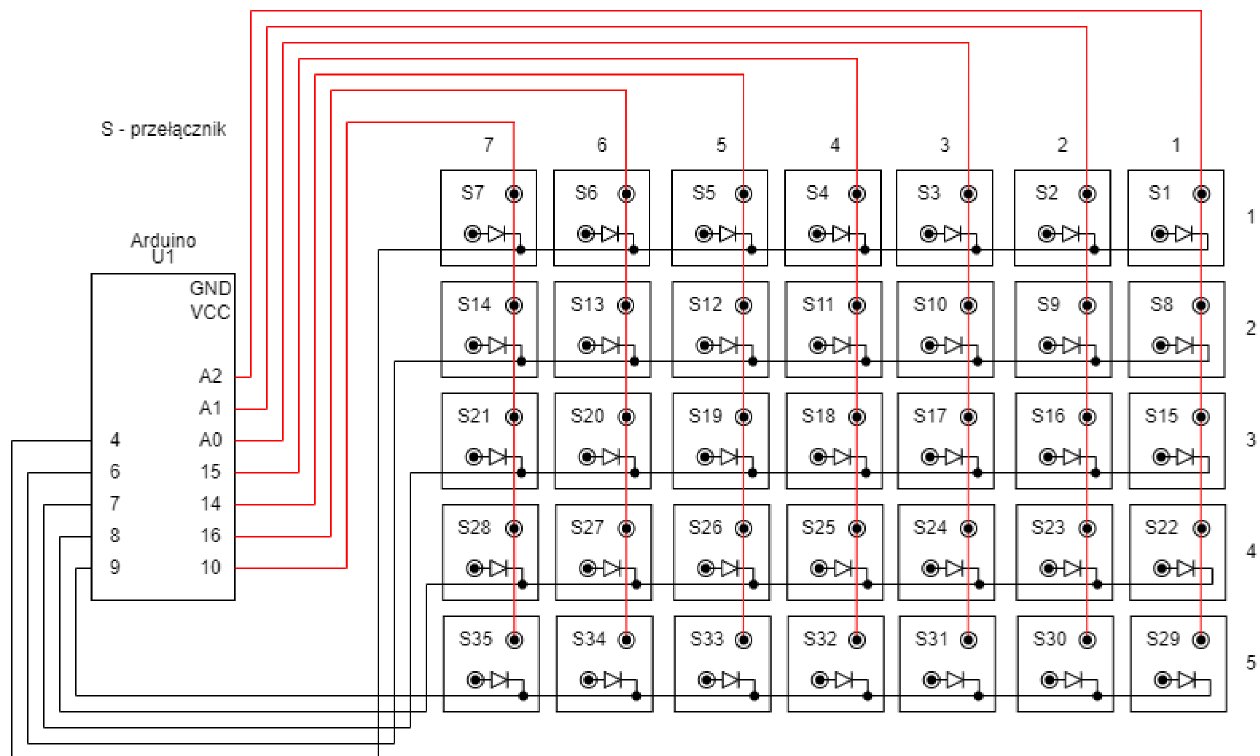
Komponent	**Ilość**
Górna obudowa	1
Dolna płyta	1
Uchwyt wyświetlacza	2

4 Schematy i modele

4.1 Podłączenie I2C



4.2 Podłączenie matrycy przełączników



5 Proces Budowy

Budowa klawiatury przebiegała według następujących kroków:

1. **Osadzenie przełączników w górnej obudowie:** Przełączniki mechaniczne zostały osadzone w górnej części obudowy wydrukowanej na drukarce 3D. Upewniono się, że wszystkie przełączniki są stabilnie zamocowane i prawidłowo wyrównane.
2. **Przygotowanie i przylutowanie do przełączników:** Każdy przełącznik został odpowiednio przygotowany poprzez przycięcie i przylutowanie diod do jednego z jego wyprowadzeń.
3. **Przycięcie, dopasowanie i izolacja przewodów:** Ocynkowane przewody zostały przycięte i dopasowane do odpowiednich długości. Izolację nałożono w miejscach, gdzie przewody mogłyby się stykać lub powodować zwarcia.
4. **Przylutowanie przełączników do przewodów:** Przełączniki zostały przylutowane do ocynkowanych przewodów, tworząc matrycę klawiszy. Połączenia te utworzyły wiersze i kolumny, które później były wykorzystywane do odczytu stanu klawiszy przez mikrokontroler.
5. **Przylutowanie przewodów do mikrokontrolera:** Przewody łączące początki kolumn i rzędów matrycy klawiszy zostały przylutowane do odpowiednich pinów mikrokontrolera Arduino Micro Pro, co umożliwia odczyt stanu klawiszy.
6. **Przylutowanie wyświetlacza OLED:** Wyświetlacz OLED został przylutowany do mikrokontrolera, a następnie zamocowany w odpowiednim miejscu w obudowie.
7. **Przygotowanie rozdziałek przewodów:** Przewody zostały przygotowane do połączenia z gniazdem TRRS. Rezystory zostały przylutowane, a przewody odpowiednio zaizolowane w miejscach, gdzie mogłyby powodować zwarcia.
8. **Przylutowanie przewodów do gniazda TRRS:** Przewody zostały przylutowane do gniazda TRRS, które następnie umieszczono w odpowiednim miejscu w obudowie, zapewniając stabilne połączenie mechaniczne i elektryczne.
9. **Wgranie programu na mikrokontroler:** Program został wgrany na mikrokontroler przy użyciu QMK Firmware. Skonfigurowano funkcje klawiszy, warstwy oraz obsługę wyświetlaczy OLED i komunikacji I2C.
10. **Montaż dolnej płyty:** Po wgraniu programu i przetestowaniu działania klawiatury, dolna płyta obudowy została przykręcona, zapewniając pełną stabilność konstrukcji.

Analogicznie wszystkie kroki zostały powtórzone dla drugiej połowy klawiatury. Jedynie kolumny drugiej połowy klawiatury zostały zostały przylutowane do pinów mikrokontrolera w odwrotnej kolejności aby program mógł je poprawnie odczytywać, prawidłowo zmapować układ który jest zdefiniowany w kodzie programu.

6 Oprogramowanie

6.1 Przygotowanie Środowiska

Zainstalowanie środowiska QMK za pomocą poradnika z oficjalnej dokumentacji `QMK Firmware`.

Skonfigurowanie środowiska za pomocą zklonowania repozytorium QMK Firmware:

```
git clone https://github.com/qmk/qmk_firmware
cd qmk_firmware
```

Instalacja zależności za pomocą QMK CLI:

```
qmk setup
```

Każda klawiatura ma swój katalog w `qmk_firmware/keyboards`. Utworzenie nowego katalogu dla swojego projektu:

```
qmk new-keyboard my_keyboard
```

6.2 Struktura Plików

Aby zaprogramować klawiaturę za pomocą QMK Firmware, konieczne jest utworzenie odpowiedniej struktury plików. Poniżej przedstawiono podstawowe pliki i foldery niezbędne do zaprogramowania klawiatury:

- **keymap.c**: Główny plik konfiguracyjny, w którym definiowane są układy klawiszy, warstwy oraz funkcje specjalne.
- **config.h**: Plik konfiguracyjny zawierający ustawienia specyficzne dla danej klawiatury, takie jak rozmiary matrycy klawiszy, piny mikrokontrolera oraz inne parametry sprzętowe.
- **rules.mk**: Plik zawierający ustawienia kompilacji, takie jak wybór mikrokontrolera, opcje kompilatora oraz aktywowane funkcje QMK.
- **readme.md**: Opcjonalny plik dokumentacyjny opisujący szczegóły budowy i konfiguracji klawiatury.

Standardowa struktura folderu projektu klawiatury w QMK wygląda następująco:

```
qmk_firmware/keyboards/
|-- my_keyboard/
|   |-- keymaps/
|   |   |-- default/
|   |       |-- keymap.c
|   |       |-- readme.md
|   |-- config.h
|   |-- rules.mk
|   |-- readme.md
```

6.3 Kod źródłowy

6.3.1 config.h

“‘

```
#pragma once
#define MATRIX_ROW_PINS { D4, D7, E6, B4, B5 }
#define MATRIX_COL_PINS { F5, F6, F7, B1, B3, B2, B6 }

#define SPLIT_WPM_ENABLE

#define MASTER_RIGHT

#define DIODE_DIRECTION COL2ROW

#define USE_I2C

#define SPLIT_OLED_ENABLE
```

“‘

- `#pragma once`: Zapobiega wielokrotnemu dołączaniu tego pliku nagłówkowego.
- `MATRIX_ROW_PINS` i `MATRIX_COL_PINS`: Definiuje piny mikrokontrolera, do których są podłączone rzędy i kolumny macierzy klawiatury.
- `SPLIT_WPM_ENABLE`: Włącza funkcję śledzenia prędkości pisania w formacie słów na minutę na podzielonej klawiaturze.
- `MASTER_RIGHT`: Ustawia prawą część klawiatury jako główną.
- `DIODE_DIRECTION COL2ROW`: Ustawia kierunek przepływu prądu przez diody, co wpływa na sposób skanowania macierzy klawiatury.
- `USE_I2C`: Włącza komunikację I^2C między częściami podzielonej klawiatury.
- `SPLIT_OLED_ENABLE`: Włącza obsługę wyświetlaczy OLED na podzielonej klawiaturze.

6.3.2 rules.mk

“‘

```
SPLIT_KEYBOARD = yes
WPM_ENABLE = yes
OLED_ENABLE = yes
```

“‘

- `SPLIT_KEYBOARD = yes`: Informuje system kompilacji, że klawiatura jest podzielona na dwie części.
- `WPM_ENABLE = yes`: Włącza funkcję śledzenia słów na minutę.
- `OLED_ENABLE = yes`: Włącza obsługę wyświetlaczy OLED.

6.3.3 keymap.c

```
#include QMK_KEYBOARD_H
enum layer_number {
    QWERTY,
    FN
};

const uint16_t PROGMEM keymaps[][MATRIX_ROWS][MATRIX_COLS] = {
    [QWERTY] = LAYOUT(
        KC_ESC,KC_1,KC_2,KC_3,KC_4,KC_5,KC_6,KC_7,KC_8,KC_9,KC_0,KC_MINS,KC_EQL,KC_BSPC,
        KC_TAB,KC_TAB,KC_Q,KC_W,KC_E,KC_R,KC_T,KC_Y,KC_U,KC_I,KC_O,KC_P,KC_LBRC,KC_RBRC,
        KC_CAPS,KC_CAPS,KC_A,KC_S,KC_D,KC_F,KC_G,KC_H,KC_J,KC_K,KC_L,KC_SCLN,KC_QUOT,
        KC_ENT,KC_LSFT,KC_LSFT,KC_Z,KC_X,KC_C,KC_V,KC_B,KC_N,KC_M,KC_COMM,KC_DOT,KC_SLSH,
        KC_BSLS,KC_RSFT,KC_LCTL,KC_LGUI,MO(1),KC_LALT,KC_SPC,MO(1),KC_ENT,KC_ENT,KC_BSPC,
        KC_SPC,KC_RALT,KC_RGUI,MO(1),KC_RCTL),

  [FN] =  LAYOUT(
        _______, _______, KC_F1, KC_F2, KC_F3, KC_F4, KC_F5, KC_F6, KC_F7, KC_F8, KC_F9,
        KC_F10, KC_F11, KC_F12, _______, _______, _______, _______, _______, _______,
        _______, _______, _______, KC_UP, _______, _______, _______, _______, _______,
        _______, _______, _______, _______, _______, _______, _______, KC_LEFT, KC_DOWN,
        KC_RGHT, _______, _______, _______, _______, _______, _______, _______, _______,
        _______, _______, _______, _______, _______, _______, _______, _______, _______,
        _______, _______, _______, _______, _______, _______, _______, _______, _______,
        _______, _______, _______, _______, _______
    )
};

static void render_logo(void) {
    static const char PROGMEM qmk_logo[] = {
        0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89, 0x8A, 0x8B, 0x8C,
        0x8D, 0x8E, 0x8F, 0x90, 0x91, 0x92, 0x93, 0x94, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4,
        0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xAD, 0xAE, 0xAF, 0xB0, 0xB1,
        0xB2, 0xB3, 0xB4, 0xC0, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, 0xC8, 0xC9,
        0xCA, 0xCB, 0xCC, 0xCD, 0xCE, 0xCF, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0x00
    };

    oled_write_P(qmk_logo, false);
}

bool oled_task_user(void) {
    render_logo();
    if(is_keyboard_master()) {
        oled_scroll_left();
    }
    return false;
}
```

- `enum layer_number`: Definiuje numerację warstw klawiatury.
- `const uint16_t PROGMEM keymaps[][MATRIX_ROWS][MATRIX_COLS]`: Definiuje mapowanie klawiszy dla poszczególnych warstw.
- `QWERTY`: Warstwa podstawowa, zawiera standardowe mapowanie klawiszy.

- `FN`: Warstwa funkcji, zawiera dodatkowe funkcje takie jak klawisze funkcyjne (F1-F12) oraz strzałki.
- `static void render_logo(void)`: Funkcja rysująca logo QMK na wyświetlaczu OLED.
- `qmk_logo`: Zawiera dane graficzne logo w formacie PROGMEM.
- `bool oled_task_user(void)`: Funkcja wywoływana przy każdym cyklu aktualizacji OLED.
- `render_logo()`: Rysuje logo na OLED.
- `if(is_keyboard_master())`: Jeśli aktualna część klawiatury jest główną (master), uruchamia przewijanie OLED w lewo.

6.4 Proces Programowania

1. **Kompilacja:** Używając QMK Firmware, skompiluj kod źródłowy klawiatury. Można to zrobić poleceniem:

```
qmk compile -kb <nazwa_klawiatury> -km <nazwa_mapy_klawiszy>
```

2. **Wprowadzenie Arduino w tryb bootloadera:** Aby wprowadzić mikrokontroler Arduino Pro Micro w tryb bootloadera, połącz pin RST z pinem GND.
3. **Wgranie programu:** Po wprowadzeniu mikrokontrolera w tryb bootloadera, użyj QMK Toolbox lub avrdude do wgrania skompilowanego pliku .hex na obie połowy klawiatury. Upewnij się, że każda połowa klawiatury jest poprawnie zaprogramowana.

7 Podsumowanie

7.1 Napotkane trudności

W trakcie pracy nad projektem napotkałem szereg trudności, które wymagały dodatkowego nakładu pracy i rozwiązań:

- **Problemy kompilacji programu** Na początku spotkałem się z trudnościami związanymi z kompilacją programu. Występowały różne błędy kompilacji, które musiałem rozwiązać, analizując komunikaty błędów i wprowadzając odpowiednie poprawki w kodzie. Było to czasochłonne i wymagało szczegółowego zrozumienia języka programowania oraz używanych bibliotek.
- **Oba wyświetlacze wyświetlają się ten sam obraz** Pomimo licznych prób, nie udało mi się rozwiązać problemu z wyświetlaniem różnych treści na dwóch wyświetlaczach. Na obu wyświetlaczach pojawiały się te same informacje, co było niezgodne z założeniami projektu. Problem ten wymagałby dalszych badań i ewentualnej modyfikacji kodu lub sprzętu.
- **Różnica rozmiaru mikrokontrolerów** W trakcie montażu okazało się, że używane przeze mnie Arduino z USB-C ma nieco większe wymiary niż standardowe wersje z micro-USB. Powodowało to problemy z dopasowaniem płytki do zaprojektowanej obudowy. Konieczne było dostosowanie obudowy lub znalezienie alternatywnego rozwiązania, aby zmieścić wszystkie komponenty.
- **Duża ilość przewodów i delikatny charakter wnętrzności klawiatury - potencjalne uszkodzenie** Projektowanie i montaż klawiatury wymagały użycia dużej liczby przewodów, co sprawiało, że wnętrze urządzenia stało się skomplikowane i chaotyczne. Delikatny charakter połączeń zwiększał ryzyko uszkodzenia podczas manipulacji i montażu. Konieczna była wyjątkowa ostrożność, aby nie uszkodzić żadnych komponentów, co dodatkowo wydłużało czas realizacji projektu.

Pomimo napotkanych trudności, udało się zrealizować wiele założeń projektowych. Klawiatura jest funkcjonalna, jednak niektóre problemy, takie jak wyświetlanie tych samych informacji na obu wyświetlaczach OLED oraz dopasowanie Arduino z USB-C do obudowy, pozostają nierozwiązane. Projekt wymaga dalszych badań i modyfikacji, aby w pełni spełnić wszystkie założenia i oczekiwania.



Rysunek 1: Finalna postać projektu

8 Źródła

1. https://pl.wikipedia.org/wiki/I^{2}C
2. https://yarukizerogames.com/wp-content/uploads/2021/09/handwired-keyboard-guide.pdf
3. https://docs.qmk.fm